

ISIS-II PL/M-86 V2.0 COMPILATION OF MODULE EXPRES
 OBJECT MODULE PLACED IN EXPR.OBJ

COMPILER INVOKED BY: :FO: EXPR.PLM DEBUG OPTIMIZE(2) DATE(FEBRUARY 2, 1982) XREF

```
1      $title ('EXPRESSION MODULE')
      expres:
      do;
```

/*

```
      modified 4/8/81 R. Silberstein
      modified 4/24/81 R. Silberstein
      modified 8/19/81 R. Silberstein
```

*/

/*

This is the module to evaluate expressions and
 instruction operands. The entry subroutines are:

```
      EXPRESSION (resultfield) byte
      OPERAND      byte
```

The expression subroutine evaluates a numeric or
 memory expression. The 'operand' routine evaluates
 a single instruction operand. Both routines return
 FALSE if an error is found, otherwise true.

```
*/
#include (:f1:macro.lit)
= $nolist
#include (:f1:expr.x86)
= $nolist
#include (:f1:ermmod.ext)
= $nolist
#include (:f1:exglob.ext)
= $nolist
$INCLUDE (:f1:SUBR2.EXT)
= $nolist
```

CP/M-86 v.1.1 (vol. IV)

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # 081-162

```
$eject
/***** global variables: *****/
```

```
69  1  dcl
      maxlev      lit '5', /* max no of nested parenthesis */
      parlevel    byte,   /* current no of parenthesis level */
      stck(600)   byte,   /* local stack within module */
      savestack   addr,   /* save of initial entry stack */
      expresser   byte,   /* error flag */
      noforward   byte,   /* true if undefined symbols to be neglected */
      bracketlegal byte,  /* true if bracket expression is legal */
      udefflag    byte;   /* true if an undefined element found */
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

$object
$include (:f1:bnf.tex)
=
= /***** 'BNF'-expression syntax *****/
= /*
= E ::= E xor A !! E or A !! A
= A ::= A and N !! N
= N ::= not N !! R
= R ::= P eq P !! P lt P !! P le F !! P gt P !! P ge P !! P ne P !! P
= P ::= P + T !! P - T !! T
= T ::= T * M !! T / M !! T mod M !! T shl M !! T shr M !! M
= M ::= - M !! + M !! S
= S ::= <over>: F !! F
= F ::= F ptr B !! seg B !! offset B !! type B !!
=      length B !! last B !! B
= B ::= ( E ) !! [ bracket-expression ] !! I
= I ::= variable !! . number !! number !! label !! string
= <over> ::= segment register
= (stringlength < 3)
=
= */

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

```

$object
/***** miscellaneous subroutines: *****/

```

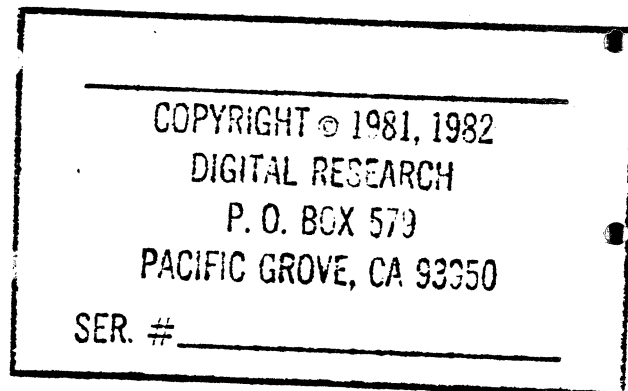
```

70 1  expexit: proc (dummy);
71 2      dcl dummy byte;
72 2      stackptr=savestack;
73 2  end expexit;

74 1  errorexit: proc;      /* return if wrong syntax */
75 2      dcl dummy byte at (.udefflag);
76 2      expresserr=false;
77 2      call expexit(dummy);
78 2  end errorexit;

79 1  clearoperand: proc(p);
80 2      dcl p address, oper based p operandstruc;
81 2      CALL FILL (0, .OPER.BASEINDEX - .OPER + 1, P);
82 2      OPER.BASEINDEX = NOOVERRIDEBIT;
83 2  end clearoperand;

```



```

/* routine to test if current token is member of a given
   set of special characters.
   Entry parameters: base  = exitvalue if token is 1. member of set
                    numbel = no of elements in set
                    pt     = pointer to list of elements
   Exit value:      routine= 0ffh if token not member of list
                    routine= base+i if token is element i,
                        token is skipped */

```

```

84 1  specmember: proc (base,numbel,pt) byte;
85 2      dcl (base,numbel,i) byte,pt address,list based pt (1) byte;
86 2      i=0ffh;
87 2      do while (i:=i+1) < numbel;
88 3          if specialtoken(list(i)) then$do call scan; return base+i; end$if;
89 3      end$while;
90 2      return 0ffh;
91 2  end specmember;

```

```

/* Routine to test if current token is member of a given set of
   operators.
   Entry/exit : see "specmember" header */

```

```

96 1  opmember: proc(base,numbel,pt) byte;
97 2      dcl (base,numbel,i,byteval) byte,pt address,list based pt (1) byte;
98 2      if token.type = operator then$do
99 3          i=0ffh;
100 3          do while (i:=i+1) < numbel;
101 4              byteval=token.value;
102 4              if byteval=list(i) then$do call scan; return base+i; end$if;
103 4          end$while;
104 3      end$if;
105 2      return 0ffh;
106 2  end opmember;

```

```

/* test if both operands are numbers, if not, error */

```

```

112 1  numbtest: proc (ptl,ptr);

```

```

113 2      dcl (ptl,ptr) address,(left based ptl,righth based ptr) operandstruc;
114 2      if (left.stype <> number) or (righth.stype <> number) then
115 2          call errexit;
116 2      end numbtest;

```

```

/* find resulting symbol type as result of an addition or a
   subtraction, test if illegal types */

```

```

117 1      typefind: proc (ptl,ptr);
118 2          dcl (ptl,ptr) address,stype byte,
              (left based ptl,righth based ptr) operandstruc;
119 2          dcl err lit '07fh',
              crosstab(9) byte data(number,variable,lab,variable,err,err,
                                      lab,err,err);

120 2          typeno: proc(typ) byte;
121 3              dcl typ byte;
122 3              if typ=number then return 0;
124 3              if typ=variable then return 1;
126 3              if typ=lab then return 2;
128 3              call errexit; /* illegal member of expression */
129 3          end typeno;

```

```

130 2          stype=crosstab(typeno(left.stype)*3+typeno(righth.stype));
131 2          if stype=err then call errexit;
133 2          left.length=left.length+righth.length;
134 2          left.stype=stype;
135 2      end typefind;

```

```

/* take care of segment specification in front of variables
   syntax: <over>: variable, <over>=ES/SS/DS/CS      */

```

```

136 1      segover: proc(pt) byte;
137 2          dcl pt address,segreg based pt byte;
138 2          if (token.type=reg) and (token.descr=dword) then$do
140 3              if nextch=':' then$do
142 4                  segreg=token.value;
143 4                  segreg=(shl(segreg,segtypecount) and segtypebit) or segbit;
144 4                  call scan; /* skip segment register */
145 4                  call scan; /* skip : */
146 4                  return 0;
147 4              end$if;
          end$if;
149 2          return 0fff;
150 2      end segover;

```

```

/* create a number operator */

```

```

151 1      createnumber: proc(p,n);
152 2          dcl p address,n addr,oper based p operandstruc;
153 2          call clearoperand(.oper);
154 2          oper.stype=number;
155 2          oper.offset=n;
156 2      end createnumber;

```

```

/* get current identifier, perform symboltable lookup
   set undefined-symbol-flag if symbol not defined,
   treat undefined symbols as numbers      */

```

```

157 1      finditem: proc (pt);
158 2          dcl pt address,left based pt operandstruc,symbptr address,i byte;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 570
 PACIFIC GROVE, CA 93950

SER. # _____

```

159 2      if token.type <> ident then$do
161 3          call clearoperand(.left);
162 3          left.stype=token.type;
163 3          left.sflag=token.descr;
164 3          left.offset=token.value;
165 3      else$do
167 3          if findsymbol(acclen,.accum(0),.symptr) then$do
169 4              call getattributes(symptr,.left);
170 4              i=left.stype;
171 4              if (i=neglected) or (i=doubleddefined) or (i=udefsymb) then$do
173 5                  udefflag=true;
174 5                  left.stype=number;
175 5                  expresserr=false;
176 5                  call errmsg(udefsymbol);
177 5              end$if;
            else$do

                /* symbol undefined - test if it is to be "neglected" */
180 4          expresserr=false;
181 4          if noforward then$do
183 5              if not newsymbol(acclen,.accum,.symptr) then$do
185 6                  call errexit;
186 6              end$if;
187 5              left.stype=neglected;
188 5              call enterattributes(symptr,.left);
189 5              end$if;

190 4          call errmsg(udefsymbol);
191 4          udefflag=true;
192 4          end$if;
193 3      end$if;
194 2      call scan;
195 2      end finditem;

/* recognize the different symboltypes for the II (identicator)
   subroutine */
196 1      syntyp: proc(pt) byte;
197 2          dcl pt address, left based pt operandstruc,i byte;
198 2          if specialtoken('&') then return 0;
200 2          if specialtoken('.') then return 1;
202 2          if token.type=string then$do
204 3              if (acclen > 0) and (acclen < 3 ) then return 2;
206 3              return 4;          /* error */
207 3          end$if;
208 2          call finditem(.left);
209 2          i=left.stype;
210 2          if (i=pseudo) or (i=operator) or (i=spec) then return 4; /* error */
212 2          return 3;
213 2      end syntyp;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

\$eject

/****** subroutines for each 'NON-TERMINAL' *****/
 /****** in 'BNF' syntax *****/

```

214 1  II: proc (pt) reentrant;
215 2    dcl pt address, left based pt operandstruc,
        doublebyt addr at (.accum(0)), saveb byte;
216 2    do case syntyp(.left);

217 3      do; /* $ */
218 4        left.stype=lab;
219 4        left.sflag=wrđ;
220 4        left.offset=cip;
221 4        if csegspec then$do /* pick up current segment specification */
223 5          left.sflag=shl(csegtyp, segtypecount) or segmbit or wrđ;
224 5          left.segment=csegvalue;
225 5        end$if;
226 4        call scan; /* skip $ */
227 4      end;

228 3      do; /* . number */
229 4        call scan; /* skip . */
230 4        call finditem(.left);
231 4        if left.stype <> number then call errexit;
233 4        left.stype=variable;
234 4        left.segment=curdseg;
235 4        left.sflag=shl(rds, segtypecount) and segtypebit;
236 4        if dspec then left.sflag=left.sflag or segmbit;
238 4      end;

239 3      do; /* string */
240 4        if acclen=1 then$do
242 5          call createnumber(.left, accum(0));
243 5        else$do
245 5          saveb=accum(0);
246 5          accum(0)=accum(1);
247 5          accum(1)=saveb;
248 5          call createnumber(.left, doublebyt);
249 5        end$if;
250 4        call scan; /* skip string */
251 4      end;

252 3      do; end; /* number, label, variable, register */
254 3      call errexit;
255 3    end$case;
256 2  end II;

257 1  BB: proc (pt) reentrant;
258 2    dcl pt address, left based pt operandstruc;
259 2    if specialtoken('(') then$do
261 3      if (parlevel:=parlevel+1) > maxlev-1 then call errexit;
263 3      call scan;
264 3      call EE(.left);
265 3      if not specialtoken(')') then call errexit;
267 3      parlevel=parlevel-1;
268 3      call scan;

```

COPYRIGHT © 1981, ILS2
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```

269 3      return;
270 3      end$if;
271 2      if specialtoken(leftbracket) then$do
273 3          if not bracketlegal then call errexit;
275 3          bracketlegal=false;
276 3          call scan; /* skip leftbracket */
277 3          call clearoperand(.left);
278 3          left.stype=number;
279 3          if not bracketexpr(.left) then call errexit;
281 3          return;
282 3      end$if;
283 2      call ll(.left);
284 2      end BB;

285 1      FF: proc (pt) reentrant;
286 2          dcl pt address, left based pt operandstruc, righ operandstruc,
                opertyp byte, val addr;
287 2          if (opertyp:=opmember(0,5,..(oseg,ooffset,otype,olength,olast)))
                                <> Offh then$do
289 3              call BB(.left);
290 3              do case opertyp;

291 4                  do; /* SEG */
292 5                      if (left.sflag and segabit) = 0 then call errexit;
294 5                      call createnumber(.left,left.segment);
295 5                      end;

296 4                  do; /* OFFSET */
297 5                      call createnumber(.left,left.offset);
298 5                      end;

299 4                  do; /* TYPE */
300 5                      call createnumber(.left,left.sflag and typebit);
301 5                      end;

302 4                  do; /* LENGTH */
303 5                      call createnumber(.left,left.length);
304 5                      end;

305 4                  do; /* LAST */
306 5                      if (val:=left.length) = 0 then val=1;
308 5                      call createnumber(.left,val-1);
309 5                      end;

310 4              end$case;
311 3          else$do
313 3              call BB(.left);
314 3              do while opmember(0,1,..(optr)) <> Offh;
315 4                  call BB(.righ);
316 4                  left.stype=righ.stype;
317 4                  left.segment=righ.segment;
318 4                  left.offset=righ.offset;
319 4                  left.baseindex=righ.baseindex;
320 4                  left.sflag=(left.sflag and typebit) or (righ.sflag and
                                (not typebit));
321 4              end$while;
322 3          end$if;

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____


```

323  2      end FF;

324  1      SS: proc (pt) reentrant;
325  2          dcl pt address, left based pt operandstruc, segreg byte;
326  2          if segover(.segreg) <> 0ffh then$do
328  3              call FF(.left);
329  3              left.sflag=(left.sflag and (not segtypebit)) or segreg;
330  3              left.baseindex=left.baseindex and (not nooverridebit);
331  3          else$do
333  3              call FF(.left);
334  3          end$if;
335  2      end SS;

336  1      MM: proc (pt) reentrant;
337  2          dcl p address, left based pt operandstruc, opertyp byte;
338  2          if (opertyp:=specmember(0,2,('+-')) <> 0ffh then$do
340  3              call MM(.left);
341  3              call numtest(.left,.left);
342  3              if opertyp=1 then$do
344  4                  left.offset=-left.offset;
345  4              end$if;
346  3          else$do
348  3              call SS(.left);
349  3          end$if;
350  2      end MM;

351  1      TT: proc (pt) reentrant;
352  2          dcl pt address, left based pt operandstruc, righth operandstruc,
353  2              opertyp byte, (leftval,righthval) addr;
354  2          call MM(.left);
355  2          do while (opertyp:=specmember(0,2,('*/')) and
356  2              opmember(2,3,('omod,oshl,oshr'))) <> 0ffh;
357  3              call MM(.righth);
358  3              call numtest(.left,.righth);
359  3              leftval=left.offset;
360  3              righthval=righth.offset;
361  3              do case opertyp;
362  4                  leftval=leftval*righthval;
363  4                  leftval=leftval/righthval;
364  4                  leftval=leftval mod righthval;
365  4                  if righthval>0 and righthval<16 then leftval=shl(leftval,righthval);
366  4                  if righthval>0 and righthval<16 then leftval=shr(leftval,righthval);
367  3              end$case;
368  3              left.offset=leftval;
369  3          end$while;
370  2      end TT;

371  1      PP: proc (pt) reentrant;
372  2          dcl pt address, left based pt operandstruc, righth operandstruc,
373  2              opertyp byte;
374  2          call TT(.left);
375  2          do while (opertyp:=specmember(0,2,('+-')) <> 0ffh;
376  3              call TT(.righth);
377  3              call typefind(.left,.righth);
378  3              if opertyp=0 then$do
379  4                  left.offset=left.offset+righth.offset;
380  4              else$do

```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 570

PACIFIC GROVE, CA 93950

SER. =

```

382 4      left.offset=left.offset-rigth.offset;
383 4      endif;
384 3      endwhile;
385 2      end PP;

386 1      RR: proc (pt) reentrant;
387 2      dcl pt address, left based pt operandstruc, rigth operandstruc,
          opertyp byte, (leftval, rigthval) addr;
388 2      call PP(.left);
389 2      if (opertyp:=opmember(0,6,.(oeq,olt,ole,ogt,oge,one))) <> Offh
                                                    then$do

391 3          call PP(.rigth);
392 3          call numtest(.left,.rigth);
393 3          leftval=left.offset;
394 3          rigthval=rigth.offset;
395 3          do case opertyp;
396 4              leftval = (leftval = rigthval);
397 4              leftval = (leftval < rigthval);
398 4              leftval = (leftval <= rigthval);
399 4              leftval = (leftval > rigthval);
400 4              leftval = (leftval >= rigthval);
401 4              leftval = (leftval <> rigthval);
402 4          end$case;
403 3          IF LEFTVAL = OFFH THEN LEFTVAL = OFFFH;
405 3          left.offset=leftval;
406 3      endif;
407 2      end RR;

408 1      NN: proc (pt) reentrant;
409 2      dcl pt address, left based pt operandstruc;
410 2      if opmember(0,1,.(onot)) <> Offh then$do
412 3          call NN(.left);
413 3          call numtest(.left,.left);
414 3          left.offset=not left.offset;
415 3      else$do
417 3          call RR(.left);
418 3      endif;
419 2      end NN;

420 1      AA: proc (pt) reentrant;
421 2      dcl pt address, left based pt operandstruc, rigth operandstruc;
422 2      call NN(.left);
423 2      do while opmember(0,1,.(oand)) <> Offh;
424 3          call NN(.rigth);
425 3          call numtest(.left,.rigth);
426 3          left.offset=left.offset and rigth.offset;
427 3      endwhile;
428 2      end AA;

429 1      EE: proc (pt) reentrant;
430 2      dcl pt address, left based pt operandstruc, right operandstruc,
          opertype byte;

431 2      call AA(.left);
432 2      do while (opertype:=opmember(0,2,.(oor,oxor))) <> Offh;
433 3          call AA(.right);
434 3          call numtest(.left,.right);

```

COPYRIGHT © 1981, 1982
 DIGITAL RESEARCH
 P. O. BOX 579
 PACIFIC GROVE, CA 93950

SER. # _____

```
435 3      if opertype=0 then$do
437 4          left.offset=left.offset or right.offset;
438 4      elseif$do
440 4          left.offset=left.offset xor right.offset;
441 4      endif;
442 3      endwhile;
443 2      end EE;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. = _____

```
reject
/***** MAIN SUBROUTINES *****/
```

```
444 1   realexpress: proc(pt);
445 2       dcl pt address,oper based pt operandstruc,
           dummy byte at(.udefflag);
446 2   savestack=stackptr; /* use local stack for reentrant routines */
447 2       stackptr=.stck(length(stck));
448 2       call EE(.oper);
449 2       call expexit(dummy);
450 2   end realexpress;
```

```
451 1   express: proc(pt) byte;
452 2       dcl pt address,oper based pt operandstruc;
453 2       expresserr=true;
454 2       udefflag=false;
455 2       parlevel=0;
456 2       call realexpress(.oper);
457 2       if udefflag then$do
458 3           oper.stype=number;
459 3           oper.sflag=byt;
460 3           oper.offset=0;
461 3       end$if;
462 3       return expresserr;
463 2   end express;
```

```
/* normal expression */
```

```
465 1   expression: proc (pt) byte public;
466 2       dcl pt address;
467 2       noforward=false;
468 2       bracketlegal=false;
469 2       return express(pt);
470 2   end expression;
```

```
/* special expression - mark all undefined symbols as 'neglected' */
```

```
471 1   noforwardexpr: proc(pt) byte public;
472 2       dcl pt address;
473 2       noforward=true;
474 2       bracketlegal=false;
475 2       return express(pt);
476 2   end noforwardexpr;
```

```
477 1   OPERND: PROC BYTE;
478 2       dcl exitvalue b,le,pt address,oper based pt operandstruc;
```

```
479 2       pt=.operands(nooper);
480 2       exitvalue=true;
481 2       bracketlegal=true;
482 2       exitvalue=express(pt);
483 2       if specialtoken(leftbracket) then$do
484 3           if bracketlegal then$do
485 4               call scan;
486 4               exitvalue=exitvalue and bracketexpr(pt);
487 4           else$do
488 4               exitvalue=false;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____

```
492 4      end$if;
493 3      end$if;
494 2      return exitvalue;
495 2      END OPERND;

496 1      OPERAND: PROC BYTE PUBLIC;
497 2          NOFORWARD = FALSE;
498 2          RETURN OPERND;
499 2      END OPERAND;

500 1      NOFORWARDOPER: PROC BYTE PUBLIC;
501 2          NOFORWARD = TRUE;
502 2          RETURN OPERND;
503 2      END NOFORWARDOPER;

504 1      end$module expres;
```

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES
------	------	------	----------------------------------

420	07EBH	70	AA PROCEDURE REENTRANT STACK=0096H 431 433
32	0000H	1	ACCLN BYTE EXTERNAL(16) 167 183 204 240
32	0000H	80	ACCUM BYTE ARRAY(80) EXTERNAL(17) 167 183 215 242 245 246 247
2			ADDR LITERALLY 10 13 16 19 32 34 40 58 61 64 67
			69 80 113 118 152 158 197 215 258 286 325 337 352 372
			387 409 421 430 445 452 478
48	0000H		ALPHANUMERIC . . . PROCEDURE BYTE EXTERNAL(25) STACK=0000H
51	0000H		ASCIICHAR PROCEDURE BYTE EXTERNAL(26) STACK=0000H
96	0008H	1	BASE BYTE PARAMETER AUTOMATIC 97 106
84	0008H	1	BASE BYTE PARAMETER AUTOMATIC 85 91
32	0008H	1	BASEINDEX BYTE MEMBER(OPERANDS)
452	0008H	1	BASEINDEX BYTE MEMBER(OPER)
133	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
352	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
113	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
113	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
152	0008H	1	BASEINDEX BYTE MEMBER(OPER)
215	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
352	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
478	0008H	1	BASEINDEX BYTE MEMBER(OPER)
430	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
372	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
421	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
387	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
421	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
430	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
409	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
286	0008H	1	BASEINDEX BYTE MEMBER(RIGHT) 319
372	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
258	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
337	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
118	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
236	0008H	1	BASEINDEX BYTE MEMBER(LEFT) 319
445	0008H	1	BASEINDEX BYTE MEMBER(OPER)
325	0008H	1	BASEINDEX BYTE MEMBER(LEFT) 330
387	0008H	1	BASEINDEX BYTE MEMBER(RIGHT)
113	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
80	0008H	1	BASEINDEX BYTE MEMBER(OPER) 81 82
197	0008H	1	BASEINDEX BYTE MEMBER(LEFT)
7			BASEREGBIT LITERALLY
7			BASEREGCOUNT LITERALLY
257	03B0H	143	BB PROCEDURE REENTRANT STACK=0026H 289 313 315
26	0000H		BRACKETEXPR PROCEDURE BYTE EXTERNAL(6) STACK=0000H 279 438
69	0261H	1	BRACKETLEGAL BYTE 273 275 468 474 481 485
7			BREGBIT LITERALLY
7			BREGCOUNT LITERALLY
4			BYT LITERALLY 460
97	0265H	1	BYTEVAL BYTE 102 103
45	0000H	1	CH BYTE PARAMETER 46

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. # _____

295	013FH	247	FF	PROCEDURE REENTRANT STACK=0038H	328	333														
36	0000H		FILEABORT.	PROCEDURE EXTERNAL(21) STACK=0000H																
39	0000H		FILL	PROCEDURE EXTERNAL(22) STACK=0000H	81															
157	01A4H	200	FINDITEK	PROCEDURE STACK=0012H	208	230														
12	0000H		FINDSYMBOL	PROCEDURE BYTE EXTERNAL(1) STACK=0000H	167															
2			FOREVER.	LITERALLY																
2			FORMFEED	LITERALLY																
15	0000H		GETATTRIBUTES.	PROCEDURE EXTERNAL(2) STACK=0000H	169															
60	0000H		HEX1OUT.	PROCEDURE EXTERNAL(29) STACK=0000H																
63	0000H		HEX2OUT.	PROCEDURE EXTERNAL(30) STACK=0000H																
97	0264H	1	I.	BYTE 100 101 103 106																
158	0267H	1	I.	BYTE 170 171																
85	0263H	1	I.	BYTE 86 87 88 91																
197	0268H	1	I.	BYTE 209 210																
3			IDENT.	LITERALLY 159																
214	02CCH	228	II	PROCEDURE REENTRANT STACK=0020H	283															
7			INDEXREGBIT.	LITERALLY																
7			INDEXREGCOUNT.	LITERALLY																
2			INIT	LITERALLY																
7			IREGBIT.	LITERALLY																
7			IREGCOUNT.	LITERALLY																
3			LAB.	LITERALLY 119 126 218																
158	0000H	9	LEFT	STRUCTURE BASED(PT)	161	162	163	164	169	170	174	187	188							
409	0000H	9	LEFT	STRUCTURE BASED(PT)	412	413	414	417												
421	0000H	9	LEFT	STRUCTURE BASED(PT)	422	425	426													
215	0000H	9	LEFT	STRUCTURE BASED(PT)	216	218	219	220	223	224	230	231	233							
				234 235 237 242 248																
352	0000H	9	LEFT	STRUCTURE BASED(PT)	353	356	357	368												
372	0000H	9	LEFT	STRUCTURE BASED(PT)	373	376	379	382												
337	0000H	9	LEFT	STRUCTURE BASED(PT)	340	341	344	348												
250	0000H	9	LEFT	STRUCTURE BASED(PT)	264	277	278	279	283											
430	0000H	9	LEFT	STRUCTURE BASED(PT)	431	434	437	440												
337	0000H	9	LEFT	STRUCTURE BASED(PT)	388	392	393	405												
197	0000H	9	LEFT	STRUCTURE BASED(PT)	208	209														
325	0000H	9	LEFT	STRUCTURE BASED(PT)	328	329	330	333												
118	0000H	9	LEFT	STRUCTURE BASED(PTL)	130	133	134													
286	0000H	9	LEFT	STRUCTURE BASED(PT)	289	292	294	297	300	303	306	308	313							
				316 317 318 319 320																
113	0000H	9	LEFT	STRUCTURE BASED(PTL)	114															
6			LEFTBRACKET.	LITERALLY 271 483																
387	0002H	2	LEFTVAL.	WORD AUTOMATIC	393	396	397	398	399	400	401	403	404	405						
352	0002H	2	LEFTVAL.	WORD AUTOMATIC	357	360	361	362	364	366	368									
421	0000H	2	LENGTH	WORD MEMBER(RIGHT)																
325	0000H	2	LENGTH	WORD MEMBER(LEFT)																
372	0000H	2	LENGTH	WORD MEMBER(LEFT)																
409	0000H	2	LENGTH	WORD MEMBER(LEFT)																
372	0000H	2	LENGTH	WORD MEMBER(RIGHT)																
421	0000H	2	LENGTH	WORD MEMBER(LEFT)																
352	0000H	2	LENGTH	WORD MEMBER(RIGHT)																
337	0000H	2	LENGTH	WORD MEMBER(LEFT)																
332	0000H	2	LENGTH	WORD MEMBER(LEFT)																
430	0000H	2	LENGTH	WORD MEMBER(RIGHT)																
387	0000H	2	LENGTH	WORD MEMBER(LEFT)																
32	0000H	2	LENGTH	WORD MEMBER(OPERANDS)																
286	0000H	2	LENGTH	WORD MEMBER(RIGHT)																
286	0000H	2	LENGTH	WORD MEMBER(LEFT)	303	306														
430	0000H	2	LENGTH	WORD MEMBER(LEFT)																

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 170

FARMING GROVE, CA 95630

SER. #

253	0000H	2	LENGTH	WORD MEMBER(LEFT)	
215	0000H	2	LENGTH	WORD MEMBER(LEFT)	
			LENGTH	BUILTIN	447
30	0000H	2	LENGTH	WORD MEMBER(OPER)	
177	0000H	2	LENGTH	WORD MEMBER(LEFT)	
452	0000H	2	LENGTH	WORD MEMBER(OPER)	
153	0000H	2	LENGTH	WORD MEMBER(LEFT)	
152	0000H	2	LENGTH	WORD MEMBER(OPER)	
387	0000H	2	LENGTH	WORD MEMBER(RIGHT)	
118	0000H	2	LENGTH	WORD MEMBER(RIGHT)	
113	0000H	2	LENGTH	WORD MEMBER(LEFT)	
478	0000H	2	LENGTH	WORD MEMBER(OPER)	
113	0000H	2	LENGTH	WORD MEMBER(RIGHT)	
445	0000H	2	LENGTH	WORD MEMBER(OPER)	
113	0000H	2	LENGTH	WORD MEMBER(LEFT)	
45	0000H		LETTER	PROCEDURE BYTE EXTERNAL(24) STACK=0000H	
2			LF	LITERALLY	
12	0000H	1	LG	BYTE PARAMETER	13
9	0000H	1	LG	BYTE PARAMETER	10
85	0000H	1	LIST	BYTE BASED(PT) ARRAY(1)	88
97	0000H	1	LIST	BYTE BASED(PT) ARRAY(1)	103
2			LIT.	LITERALLY	2 3 4 5 6 7 8 69 119
69			MAXLEV	LITERALLY	261
336	0573H	71	MM	PROCEDURE REENTRANT STACK=0048H	340 353 355
66	0000H	2	N.	WORD PARAMETER	67
63	0000H	2	N.	WORD PARAMETER	64
39	0000H	1	N.	BYTE PARAMETER	40
57	0000H	1	N.	BYTE PARAMETER	58
60	0000H	1	N.	BYTE PARAMETER	61
151	0004H	2	N.	WORD PARAMETER AUTOMATIC	152 155
3			NEGLECTED.	LITERALLY	171 187
9	0000H		NEWSYMBOL.	PROCEDURE BYTE EXTERNAL(0) STACK=0000H	183
32	0000H	1	NEXTCH	BYTE EXTERNAL(15)	140
4			NIL.	LITERALLY	
403	07AFH	60	NN	PROCEDURE REENTRANT STACK=0086H	412 422 424
69	0260H	1	NOFORWARD.	BYTE	181 467 473 497 501
471	0906H	23	NOFORWARDEXPR. . .	PROCEDURE BYTE PUBLIC STACK=00B8H	
500	0979H	13	NOFORWARDOPER. . .	PROCEDURE BYTE PUBLIC STACK=00BAH	
32	0000H	1	NOOPER	BYTE EXTERNAL(18)	479
7			NOOVERCOUNT. . . .	LITERALLY	
7			NOOVERRIDEBIT. . . .	LITERALLY	82 330
96	0006H	1	NUMBEL	BYTE PARAMETER AUTOMATIC	97 101
84	0006H	1	NUMBEL	BYTE PARAMETER AUTOMATIC	85 87
3			NUMBER	LITERALLY	114 119 122 154 174 231 278 459
112	00B8H	28	NUMBEST	PROCEDURE STACK=0010H	341 356 392 413 425 434
6			OAND	LITERALLY	423
6			OEQ.	LITERALLY	389
452	0006H	2	OFFSET	WORD MEMBER(OPER)	461
445	0006H	2	OFFSET	WORD MEMBER(OPER)	
430	0006H	2	OFFSET	WORD MEMBER(LEFT)	437 440
421	0006H	2	OFFSET	WORD MEMBER(RIGHT)	426
421	0006H	2	OFFSET	WORD MEMBER(LEFT)	426
409	0006H	2	OFFSET	WORD MEMBER(LEFT)	414
387	0006H	2	OFFSLT	WORD MEMBER(RIGHT)	394
387	0006H	2	OFFSET	WORD MEMBER(LEFT)	393 405
372	0006H	2	OFFSET	WORD MEMBER(RIGHT)	379 382
372	0006H	2	OFFSET	WORD MEMBER(LEFT)	379 382

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

BOX 60

FARMINGTON, UTAH 84201

133

133

SER. #

352	0006H	2	OFFSET	WORD MEMBER(RIGHT)		358			
352	0006H	2	OFFSET	WORD MEMBER(LEFT)		357	368		
337	0006H	2	OFFSET	WORD MEMBER(LEFT)		344			
32	0006H	2	OFFSET	WORD MEMBER(OPERANDS)					
325	0006H	2	OFFSET	WORD MEMBER(LEFT)					
286	0006H	2	OFFSET	WORD MEMBER(RIGHT)		318			
286	0006H	2	OFFSET	WORD MEMBER(LEFT)		297	318		
478	0006H	2	OFFSET	WORD MEMBER(OPER)					
230	0006H	2	OFFSET	WORD MEMBER(LEFT)					
215	0006H	2	OFFSET	WORD MEMBER(LEFT)		220			
430	0006H	2	OFFSET	WORD MEMBER(RIGHT)		437	440		
197	0006H	2	OFFSET	WORD MEMBER(LEFT)					
158	0006H	2	OFFSET	WORD MEMBER(LEFT)		164			
152	0006H	2	OFFSET	WORD MEMBER(OPER)		155			
118	0006H	2	OFFSET	WORD MEMBER(RIGHT)					
118	0006H	2	OFFSET	WORD MEMBER(LEFT)					
30	0006H	2	OFFSET	WORD MEMBER(OPER)					
113	0006H	2	OFFSET	WORD MEMBER(RIGHT)					
113	0006H	2	OFFSET	WORD MEMBER(LEFT)					
6			OGE.	LITERALLY	389				
6			OGT.	LITERALLY	389				
6			OLAST.	LITERALLY	287				
6			OLE.	LITERALLY	389				
6			OLENGTH.	LITERALLY	287				
6			OLT.	LITERALLY	389				
6			OMOD	LITERALLY	354				
6			ONE.	LITERALLY	389				
6			ONOT	LITERALLY	410				
6			OOFFSET.	LITERALLY	287				
6			OOR.	LITERALLY	432				
452	0000H	9	OPER	STRUCTURE BASED(P)		456	459	460	461
80	0000H	9	OPER	STRUCTURE BASED(P)		81	82		
132	0000H	9	OPER	STRUCTURE BASED(P)		153	154	155	
445	0000H	9	OPER	STRUCTURE BASED(P)		448			
478	0000H	9	OPER	STRUCTURE BASED(P)					
496	096CH	13	OPERAND.	PROCEDURE BYTE PUBLIC STACK=00BAH					
32	0000H	36	OPERANDS	STRUCTURE ARRAY(4) EXTERNAL(19)			479		
7			OPERANDSTRUC	LITERALLY	32	80	113	118	152
					337	352	372	387	409
					421	430	445	452	478
3			OPERATOR	LITERALLY	98	210			
477	091DH	79	OPERND	PROCEDURE BYTE STACK=00B6H			498	502	
337	000FH	1	OPERTYP.	BYTE AUTOMATIC	389	395			
352	000FH	1	OPERTYP.	BYTE AUTOMATIC	354	359			
337	0002H	1	OPERTYP.	BYTE AUTOMATIC	338	342			
372	000BH	1	OPERTYP.	BYTE AUTOMATIC	374	377			
286	000DH	1	OPERTYP.	BYTE AUTOMATIC	287	290			
430	000BH	1	OPERTYPE	BYTE AUTOMATIC	432	435			
96	0074H	68	OPMEMBER	PROCEDURE BYTE STACK=000AH			287	314	354
6			OPTR	LITERALLY	314				
6			OSEG	LITERALLY	287				
6			OSHL	LITERALLY	354				
6			OSHORT	LITERALLY					
6			OSHR	LITERALLY	354				
6			OTYPE.	LITERALLY	287				
33	0000H		OUTTEXT.	PROCEDURE EXTERNAL(20) STACK=0000H					
6			OXOR	LITERALLY	432				
79	0004H	2	P.	WORD PARAMETER AUTOMATIC		80	81	82	

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #

151	0006H	2	P.	WORD PARAMETER AUTOMATIC	152	153	154	155				
39	0006H	1	PARLEVEL	BYTE 261 267 455								
371	0685H	90	PP	PROCEDURE REENTRANT STACK=006CH			388	391				
2			PROC	LITERALLY 9 12 15 18 21 23 26 29 48 51 60								
				63 66 70 74 79 84 96 112 117 120 136 151 157 196								
				214 257 285 324 336 351 371 386 408 420 429 444 451 465								
				471 477 496 500								
3			PSEUDO	LITERALLY 210								
465	0004H	2	PT	WORD PARAMETER AUTOMATIC	466	467						
84	0004H	2	PT	WORD PARAMETER AUTOMATIC	85	88						
451	0004H	2	PT	WORD PARAMETER AUTOMATIC	452	456	459	460	461			
429	000EH	2	PT	WORD PARAMETER AUTOMATIC	430	431	434	437	440			
420	000EH	2	PT	WORD PARAMETER AUTOMATIC	421	422	425	426				
408	0004H	2	PT	WORD PARAMETER AUTOMATIC	409	412	413	414	417			
386	0012H	2	PT	WORD PARAMETER AUTOMATIC	387	388	392	393	405			
371	000EH	2	PT	WORD PARAMETER AUTOMATIC	372	373	376	379	382			
39	0000H	2	PT	WORD PARAMETER 40								
351	0012H	2	PT	WORD PARAMETER AUTOMATIC	352	353	356	357	368			
336	0006H	2	PT	WORD PARAMETER AUTOMATIC	337	340	341	344	348			
324	0006H	2	PT	WORD PARAMETER AUTOMATIC	325	328	329	330	333			
285	0010H	2	PT	WORD PARAMETER AUTOMATIC	286	289	292	294	297 300 303 306			
				308 313 316 317 318 319 320								
257	0004H	2	PT	WORD PARAMETER AUTOMATIC	258	264	277	278	279 283			
26	0000H	2	PT	WORD PARAMETER 27								
214	0006H	2	PT	WORD PARAMETER AUTOMATIC	215	216	218	219 220 223 224 230				
				231 233 234 235 237 242 248								
196	0004H	2	PT	WORD PARAMETER AUTOMATIC	197	208	209					
157	0004H	2	PT	WORD PARAMETER AUTOMATIC	158	161	162 163 164 169 170 174					
				187 188								
444	0004H	2	PT	WORD PARAMETER AUTOMATIC	445	448						
136	0004H	2	PT	WORD PARAMETER AUTOMATIC	137	142	143					
96	0004H	2	PT	WORD PARAMETER AUTOMATIC	97	103						
478	0004H	2	PT	WORD 478 479 482 488								
471	0004H	2	PT	WORD PARAMETER AUTOMATIC	472	475						
117	0006H	2	PTL	WORD PARAMETER AUTOMATIC	118	130	133 134					
112	0006H	2	PIL	WORD PARAMETER AUTOMATIC	113	114						
117	0004H	2	PTR	WORD PARAMETER AUTOMATIC	118	130	133					
112	0004H	2	PTR	WORD PARAMETER AUTOMATIC	113	114						
5			RBP	LITERALLY								
5			RBX	LITERALLY								
5			RCS	LITERALLY								
5			RDI	LITERALLY								
5			RDS	LITERALLY 235								
444	003BH	37	REALEXPRESS	PROCEDURE STACK=00ACH	456							
2			REENT	LITERALLY								
3			REG	LITERALLY 138								
5			RES	LITERALLY								
12	0000H	2	RESULT	WORD PARAMETER 13								
9	0000H	2	RESULT	WORD PARAMETER 10								
430	0002H	9	RIGHT	STRUCTURE AUTOMATIC	433	434	437 440					
6			RIGHTBRACKET	LITERALLY								
421	0002H	9	RIGH	STRUCTURE AUTOMATIC	424	425	426					
387	0006H	9	RIGH	STRUCTURE AUTOMATIC	391	392	394					
372	0002H	9	RIGH	STRUCTURE AUTOMATIC	375	376	379 382					
352	0006H	9	RIGH	STRUCTURE AUTOMATIC	355	356	358					
236	0004H	9	RIGH	STRUCTURE AUTOMATIC	315	316	317 318 319 320					
118	0000H	9	RIGH	STRUCTURE BASED(PTR)	130	133						

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P.O. BOX 570

PACIFIC GROVE, CA 93950

SER. #

113	0000H	9	R1GTH.	STRUCTURE BASED(PTR)	114	
387	0004H	2	R1GTHVAL	WORD AUTOMATIC	394 396 397 398 399 400 401	
352	0004H	2	R1GTHVAL	WORD AUTOMATIC	358 360 361 362 363 364 365 366	
336	06DFH	208	RR	PROCEDURE REENTRANT STACK=0080H	417	
5			RSI.	LITERALLY		
5			RSS.	LITERALLY		
57	0000H	2	S.	WORD PARAMETER	58	
215	0002H	1	SAVEB.	BYTE AUTOMATIC	245 247	
69	0000H	2	SAVESTACK.	WORD	72 446	
21	0000H		SCAN	PROCEDURE EXTERNAL(4) STACK=0000H	90 105 144 145 194 226	
				229 250 263 268 276 487		
7			SEGBIT.	LITERALLY	143 223 237 292	
7			SEGMCOUNT.	LITERALLY		
452	0004H	2	SEGMENT.	WORD MEMBER(OPER)		
445	0004H	2	SEGMENT.	WORD MEMBER(OPER)		
80	0004H	2	SEGMENT.	WORD MEMBER(OPER)		
430	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
430	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
421	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
421	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
409	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
387	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
387	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
372	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
372	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
352	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
352	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
337	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
32	0004H	2	SEGMENT.	WORD MEMBER(OPERANDS)		
325	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
286	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)	317	
286	0004H	2	SEGMENT.	WORD MEMBER(LEFT)	294 317	
258	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
215	0004H	2	SEGMENT.	WORD MEMBER(LEFT)	224 234	
197	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
153	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
152	0004H	2	SEGMENT.	WORD MEMBER(OPER)		
118	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
118	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
113	0004H	2	SEGMENT.	WORD MEMBER(RIGHT)		
113	0004H	2	SEGMENT.	WORD MEMBER(LEFT)		
473	0004H	2	SEGMENT.	WORD MEMBER(OPER)		
136	0145H	63	SEGOVER.	PROCEDURE BYTE STACK=0006H	326	
325	0002H	1	SEGREG	BYTE AUTOMATIC	326 329	
137	0000H	1	SEGREG	BYTE BASED(PT)	142 143	
7			SEGTY: BIT	LITERALLY	143 235 329	
7			SEGTYPECOUNT	LITERALLY	143 223 235	
452	0003H	1	SFLAG.	BYTE MEMBER(OPER)	460	
445	0003H	1	SFLAG.	BYTE MEMBER(OPER)		
80	0003H	1	SFLAG.	BYTE MEMBER(OPER)		
430	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)		
430	0003H	1	SFLAG.	BYTE MEMBER(LEFT)		
421	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)		
421	0003H	1	SFLAG.	BYTE MEMBER(LEFT)		
409	0003H	1	SFLAG.	BYTE MEMBER(LEFT)		
387	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)		
387	0003H	1	SFLAG.	BYTE MEMBER(LEFT)		

COPYRIGHT © 1981, 1982

DIGITAL EQUIPMENT CORPORATION

FACSIMILE NO. 01-859-0000

SER. #

372	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)	
372	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
352	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)	
352	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
337	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
32	0003H	1	SFLAG.	BYTE MEMBER(OPERANDS)	
325	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	329
286	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)	320
286	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	292 300 320
238	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
215	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	219 223 235 237
197	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
158	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	163
152	0003H	1	SFLAG.	BYTE MEMBER(OPER)	
118	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)	
118	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
113	0003H	1	SFLAG.	BYTE MEMBER(RIGHT)	
113	0003H	1	SFLAG.	BYTE MEMBER(LEFT)	
478	0003H	1	SFLAG.	BYTE MEMBER(OPER)	
			SHL.	BUILTIN	143 223 235 364
			SHR.	BUILTIN	366
18	0000H	2	SOURCE	WORD PARAMETER	19
2			SPACE.	LITERALLY	
3			SPEC	LITERALLY	210
23	0000H		SPECIALTOKEN . . .	PROCEDURE BYTE EXTERNAL(5) STACK=0000H	88 198 200 259 265
				271 433	
84	003EH	54	SPECMEMBER	PROCEDURE BYTE STACK=000CH	338 354 374
324	0536H	61	SS	PROCEDURE REENTRANT STACK=0040H	348
			STACKPTR	BUILTIN	72 446 447
69	0007H	600	STCK	BYTE ARRAY(600)	447
12	0000H	2	STRADR	WORD PARAMETER	13
9	0000H	2	STRADR	WORD PARAMETER	10
3			STRING	LITERALLY	202
2			STRUC.	LITERALLY	32 80 113 118 152 158 197 215 258 286 325
				337 352 372 387 409 421 430 445 452 478	
452	0002H	1	STYPE.	BYTE MEMBER(OPER)	459
445	0002H	1	STYPE.	BYTE MEMBER(OPER)	
80	0002H	1	STYPE.	BYTE MEMBER(OPER)	
430	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	
430	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
421	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	
421	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
409	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
387	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	
387	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
372	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	
372	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
352	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	
352	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
337	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
32	0002H	1	STYPE.	BYTE MEMBER(OPERANDS)	
325	0002H	1	STYPE.	BYTE MEMBER(LEFT)	
286	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	316
286	0002H	1	STYPE.	BYTE MEMBER(LEFT)	316
258	0002H	1	STYPE.	BYTE MEMBER(LEFT)	278
215	0002H	1	STYPE.	BYTE MEMBER(LEFT)	218 231 233
197	0002H	1	STYPE.	BYTE MEMBER(LEFT)	209

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 679
PACIFIC GROVE, CA 93950

SER. # _____

158	0002H	1	STYPE.	BYTE MEMBER(LEFT)	162	170	174	187	
152	0002H	1	STYPE.	BYTE MEMBER(OPER)	154				
118	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	130				
118	0002H	1	STYPE.	BYTE MEMBER(LEFT)	130	134			
118	0266H	1	STYPE.	BYTE	130	131	134		
113	0002H	1	STYPE.	BYTE MEMBER(RIGHT)	114				
113	0002H	1	STYPE.	BYTE MEMBER(LEFT)	114				
470	0002H	1	STYPE.	BYTE MEMBER(OPER)					
18	0000H	2	SYMBADR.	WORD PARAMETER	19				
15	0000H	2	SYMBADR.	WORD PARAMETER	16				
3			SYMBOL	LITERALLY					
158	0002H	2	SYMBPTR.	WORD	167	169	183	188	
196	026CH	96	SYNTYP	PROCEDURE BYTE STACK=0018H				216	
33	0000H	2	T.	WORD PARAMETER	34				
2			TAB.	LITERALLY					
36	0000H	2	TEXTADR.	WORD PARAMETER	37				
2			THENDO	LITERALLY	88	98	103	138	140
					221	240	259	271	287
					326	338	342	377	389
					410	435	457	483	
					485				
23	0000H	1	TOK.	BYTE PARAMETER	24				
32	0000H	4	TOKEN.	STRUCTURE EXTERNAL(14)		98	102	138	142
						159	162	163	164
					202				
2			TRUE	LITERALLY	173	191	453	473	480
351	05BAH	203	TT	PROCEDURE REENTRANT STACK=005CH				373	375
120	0004H	1	TYP.	BYTE PARAMETER AUTOMATIC	121	122	124	126	
32	0000H	1	TYPE	BYTE MEMBER(TOKEN)	98	138	159	162	202
7			TYPEBIT.	LITERALLY	300	320			
7			TYPECOUNT.	LITERALLY					
117	00D4H	73	TYPEFIND	PROCEDURE STACK=0018H				376	
120	011DH	40	TYPEEND	PROCEDURE BYTE STACK=000EH				130	
69	0262H	1	UDEFFLAG	BYTE	75	173	191	445	454
3			UDEFSYMB	LITERALLY	171				
8			UDEFSYMBOL	LITERALLY	176	190			
54	0000H		UPPER.	PROCEDURE BYTE EXTERNAL(27) STACK=0000H					
286	0002H	2	VAL.	WORD AUTOMATIC	306	307	308		
32	0002H	2	VALUE.	WORD MEMBER(TOKEN)	102	142	164		
3			VARIABLE	LITERALLY	119	124	233		
4			WRD.	LITERALLY	219	223			

MODULE INFORMATION:

CODE AREA SIZE = 0986H 2438D
 CONSTANT AREA SIZE = 0022H 34D
 VARIABLE AREA SIZE = 026AH 618D
 MAXIMUM STACK SIZE = 00BAH 186D
 801 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-86 COMPILATION

COPYRIGHT © 1981, 1982

DIGITAL RESEARCH

P. O. BOX 579

PACIFIC GROVE, CA 93950

SER. #